Urns Filled with Bitcoins:

New Perspectives on Proof-of-Work Mining

José Parra-Moyano[1], Gregor Reich[2], and Karl Schmedders[3]

[1,2,3]University of Zurich

June 5, 2019

Abstract

The probability of a miner finding a valid block in the bitcoin blockchain is assumed to follow the Poisson distribution. However, simple, descriptive, statistical analysis reveals that blocks requiring a lot of time to find—long blocks—are won only by miners with a relatively higher hash power per second. This suggests that relatively bigger miners might have an advantage with regard to winning long blocks, which can be understood as a sort of "within block learning". Modelling the bitcoin mining problem as a race, and by means of a multinomial logit model, we can reject that the time spent mining a particular block does not affect the probability of a miner finding a valid version of this block in a manner that is proportional to her size. Further, we postulate that the probability of a miner finding a valid block is governed by the negative hypergeometric distribution. This would explain the descriptive statistics that emerge from the data and be aligned with the technical aspects of bitcoin mining. We draw an analogy between bitcoin mining and the classical "urn problem" in statistics to sustain our theory. This result can have important consequences for the miners of proof-of-work cryptocurrencies in general, and for the bitcoin mining community in particular.

Acknowledgements

We would like to sincerely thank Prof. Dr. Claudio Tessone, Dr. Christian Jaag, Prof. Dr. Shahram Khazaei, and Dr. Juraj Šarinay for their comments and inputs on this paper.

1 Introduction

Bitcoin is the electronic peer-to-peer currency, payment, and economic system first proposed by Nakamoto (2008) that has since its introduction attracted the attention of scholars and practitioners. The paper by Nakamoto (2008) opened a new stream of research in the literature and gave rise to both the industry of "blockchain"—the underlying technology behind bitcoin—and "cryptocurrencies". The market capitalization of bitcoin and all existing cryptocurrencies reached 800 billion US dollars in 2018 (CoinMarketCap, 2019) and the size of the blockchain technology market is predicted to reach an annual value of more than 23 billion US dollars by 2023 (Grand View Research, 2018).

One of the properties that made bitcoin unique at the time of its introduction was the fact that it uses a system called "mining", which is conducted by machines that implement the bitcoin protocol and are called "miners" in order to reach a secure, tamper-resistant consensus with regard to the transactions made in the system, as well as to generate and introduce new bitcoins—units of the currency—into the system (Antonopoulos, 2014). Miners conduct a trial-and-error process to find a "valid" block—a piece of information that is linked to the past history of all the bitcoin transactions and that, among other things, generates a predefined number of new bitcoins. The probability of a miner finding a valid block is determined by her individual "hash rate per second" (the number of trials that she can conduct per second) and the mining difficulty, D, which is a predefined number that is computed to keep the average time required until any of the participating miners finds a valid block at 10 minutes. Each block contains a reference to all the previous blocks that have been found. This forces a chronological order on the mining process and results in the fact that miners compete simultaneously to find a valid version of the "next block". The mining process resembles a race, in which miners compete to become the first to find the next valid block. As soon as a miner finds the next valid block, the winning miner broadcasts the valid block to all the other participating miners such that a new race containing this new valid block can start. Miners can collaborate with each other by gathering in "mining pools". For the sake of our analysis we use "miner" whenever we refer to either a miner or a mining pool and "mining pool" when we refer specifically to a mining pool.

It is broadly accepted that the probability of finding a valid block—the "arrival rate" of valid blocks—is a Poisson process (Bowden, Keeler, Krzesinski, & Taylor, 2018). This is assumed by authors including Nakamoto (2008), Rosenfeld (2011), Eyal and Sirer (2013), Decker and Wattenhofer (2013), Rosenfeld (2014), A. K. Miller and LaViola (2014), Sapirshtein, Sompolinsky, and Zohar (2015), Göbel, Keeler, Krzesinski, and Taylor (2015), Lewenberg, Bachrach, Sompolinsky, Zohar, and Rosenschein (2015), Houy (2016), Cocco and Marchesi (2016), Solat and Potop-Butucaru (2016), Beccuti and Jaag (2017), Chiu and

Koeppl (2017), Dimitri (2017), Aggarwal, Brennen, Lee, Santha, and Tomamichel (2018), L. Cong, Li, and Wang (2018), Hayes (2019), Easley, O'Hara, and Basu (2019), L. W. Cong, He, and Li (2019), and Wang et al. (2019). The assumption that the mining process follows the Poisson distribution implies that the miners' probabilities of winning remain constant throughout the race (i.e., throughout the time that elapses between the moment at which a miner starts trying to find the next valid block and the moment at which any miner finds and broadcasts that valid block). This also means that two miners with identical and constant hash rates per second have identical and constant probabilities of winning throughout the race, regardless of the moment at which they start mining and regardless of the number of failed trials they have previously conducted for this particular block. Or, phrased differently, the assumption implies that the length of the race (the time that passes between the moment at which a miner starts mining a particular block and that at which a valid version of this block is found) does not influence the probability of the miner winning. This assumption is used by the Bitcoin protocol to set the mining difficulty, D, which determines the expected time required until a valid block is found. This assumption is also used by all the miners that can participate in the mining process in order to determine their expected returns on mining and hence to decide whether to participate in the mining process or not (entry–exit decision). Further, this assumption is used by cryptocurrency exchanges to estimate the hash rate of mining pools, and by cryptographers to determine the security level of all bitcoin-like (proof-of-work) blockchains. We theorize and suggest, however, that the probability of winning a block increases during the time that a miner spends mining this block. This implies that the probability of winning a block does not follow the Poisson distribution and we suggest that it could follow the negative hypergeometric distribution instead. We test our postulate by means of a multinomial logit model similar to the one used by Bolton and Chapman (1986) to model horse races and conclude that we cannot reject the hypothesis that the probability of winning a block increases along the time that a miner spends mining this block. This result implies that the probability of winning varies throughout the time that miners are mining the same block, that bigger miners have a higher probability of winning longer blocks (or races), that it might be in the interest of smaller miners to stop mining when they reach a certain time threshold, that cryptocurrency exchanges need to change the way in which they calculate mining pools' hash rates and "luck"[1], and that the way in which the Bitcoin difficulty, D, is determined needs to be adapted. Further, it implies that the expected return on mining for relatively small miners is smaller than is assumed, whereas the expected return on mining for relatively large miners is higher than is assumed. That the probability of winning a block follows the hypergeometric distribution is a plausible explanation to this phenomenon.

In Section 2 we describe the fundamentals of the mining processes of bitcoin as well as the observations

[1]Pool luck is a parameter that emerges from the difference between the expected success rate of mining (given the hash power) and the observed success rate of mining.

and assumptions made by other scholars with regard to the mining process following the Poisson distribution. Further, we propose and explain why we consider that the mining processes follows the negative hypergeometric distribution, making an analogy between bitcoin mining and the classical "urn problem" in statistics. In Section 3 we describe the data that we use and present the descriptive statistics that motivate our postulate. In Section 4 we present the multinomial logit model that we use to model the bitcoin mining process. In Section 5 we show the results of the estimation. In Section 6 we discuss the implications of our results and we conclude.

2 Bitcoin Mining

In this section we briefly describe the concept of hashing, we present the fundamentals of bitcoin mining, we describe the assumptions scholars make in order to conclude that the arrival rate of valid blocks follows the Poisson distribution, and we describe why we consider that the probability of winning a block could in fact follow the negative hypergeometric distribution.

2.1 Fundamentals of the SHA-256 Function

The SHA-256 (Secure Hash Algorithm 256) function is a cryptographic, one-way compression function. The domain of the SHA-256 function is composed by any string of length up to 2^{64} bits. This implies that the domain of the SHA-256 function contains $2^{(2^{64}-1)}$ possible, different inputs. This is the set of possible inputs to the function. The range (support) of the SHA-256 function encompasses 2^{256} possible 256-bit strings. All the strings that result from the SHA-256 function have a length of 256 bits. The result of a hashing function is called a "hash". The SHA-256 function is deterministic, such that the same input always yields the same output. Altering one bit in the input passed through a hashing function completely alters the resulting output. The SHA-256 function is a one-way function, which means that the original data can not be retrieved from the resulting data (i.e., in order to find the input that yields a particular output, only a trial-and-error process can be conducted). Since the SHA-256 function cannot be inverted, it is impossible to anticipate which input is going to yield a particular output. Courtois, Grajek, and Naik (2014a) describe further details of the SHA-256 function applied to the context of bitcoin mining.

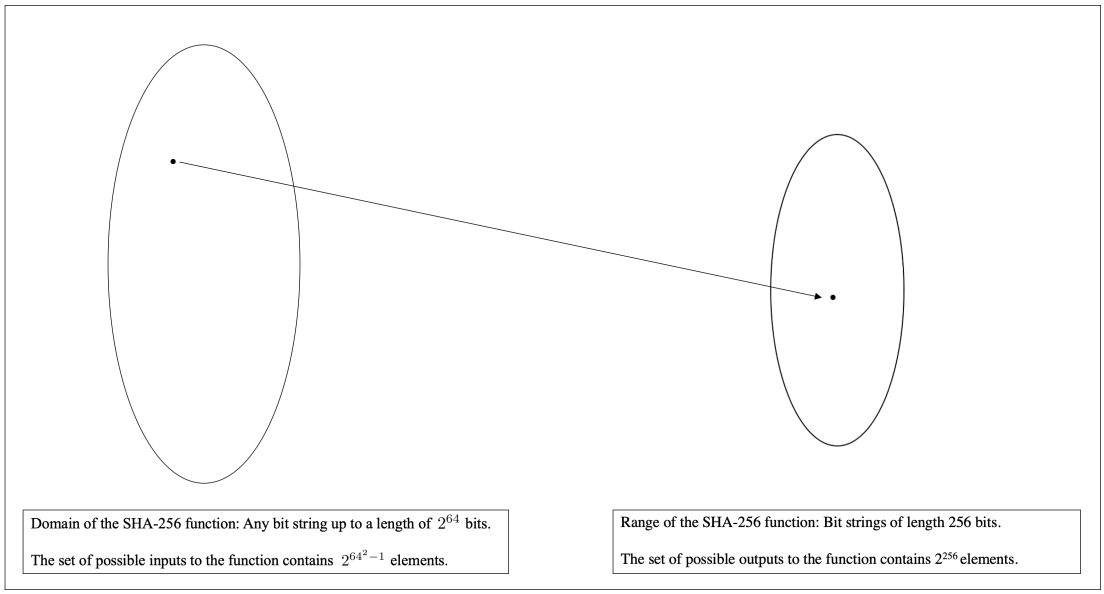

Figure 1: Illustration of the domain and range of the SHA-256 function.

2.2 Fundamentals of Bitcoin Mining

The underlying technology behind bitcoin is called the blockchain. A blockchain is a peer-to-peer ledger that stores information in packages called blocks. Every block is linked to each other block by containing the hash that results from hashing the previous block. Blocks have a size limit set to 1 MB [2]. As explained by Courtois, Grajek, and Naik (2014b), blocks contain the following information:

- **Version number**: An integer representing the version number of the bitcoin software. This number defines the rules governing the blocks.
- **The hash of the previous block**: All the input of the previous block is hashed, such that the resulting hash can be included in the current block. This links each block with the previous one.
- **The Merkle root**: A Merkle root is part of a Merkle tree and makes a reference to the transactions that are stored in this block. The Merkle root can be understood as an aggregated hash of all the transactions contained in the block. It is important to note that one of the transactions stored in the block is written by the miner writing the block itself. This transaction is the transaction that creates and assigns the new bitcoins to the miner itself writing the block. This is the way in which miners are compensated for their mining costs.
- **Timestamp**: The time at the moment at which the block was written.

[2]Due to the Segregated Witness (SegWit) protocol upgrade implemented in 2017, blocks that are larger than 1 MB are accepted if they fulfill a set of requirements. For more information on SegWit see https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

- **Target**: A global variable target, also known as Difficulty, D, that determines the blocks that will be considered "valid".
- **Padding + Len**: Two constants required by the SHA-256 hash function.
- **Nonce**: The nonce is a 32-bit number chosen by the miner.

The objective of miners is to take the inputs given by the network (the version number, the hash of the previous block, the timestamp, the target, and the Padding + Len), incorporate the Merkle root with transactions waiting to be included in the blockchain, hash it using the SHA-256 function, and hash the resulting 256-bit string again using the same SHA-256 function, such that the second resulting hash starts with a number of zeroes larger than that determined by the target (or Difficulty)[3]. As soon as a miner finds a nonce that, together with the input given by the network and the information that she has written, is hashed twice and yields a 256-bit string starting with the minimum accepted amount of zeroes, she has found a valid block. At the moment at which a miner finds a valid block, it is in her interest to broadcast this valid block to the network such that the other miners can verify that the input and the nonce provided by the winning miner, once hashed, in fact start with the desired number of zeroes and accept the broadcasted block as valid. Once the block is accepted as valid, a new problem using the hash of this newly accepted block as the one of the inputs for the next block starts for all the miners. Since the SHA-256 function cannot be inverted, it is impossible to anticipate which input (which nonce) is going to yield a particular output. This is the cornerstone of proof-of-work mining: miners have to take the input given by the previous block, add the corresponding Merkle root as well as the further required information, and try many different 32-bit random nonces such that once that they are hashed together (the nonce and the rest of the information written in the block) twice, the result of the function yields a number starting with at least a certain amount of zeroes. Should the nonces be exhausted for a potentially valid block, the miner just includes a "superNonce" in the coinbase of the Merkle tree—a field that can be written, with no format specification, by the miner—generating a new version of the potentially valid block. Then, the miner repeats the process and can try a new set of nonces for this new potentially valid block. As explained by Courtois et al. (2014a), since the size of this extraNonce is only limited by the size of block itself, it can be as large as required as long as the block size is within protocol limits. Adding the extraNonce alters the resulting hash of the block in which the miner is working without interfering with its correctness, and therefore allows the miner to try a new set of 32-bit nonces to find a valid block.

The process of mining (this trial-and-error process of testing many different nonces such that the result yields a number below a predefined threshold) is carried out by specialized hardware that consumes elec-

[3]There are methods to speed up this double-hashing procedure. See Hanke (2016) and Courtois et al. (2014b).

tricity. The power of mining hardware is measured in hashes per second (i.e., the number of hashing operations that the machine can achieve per second). Miners are motivated to mine due to the "block reward" and "block fees". The block reward is the amount of newly created bitcoins, defined by the protocol, that the miner can create and assign to herself in a transaction written in each potentially valid block. This transaction becomes effective and accepted by the network only if the potentially valid block becomes effectively valid. Note that the transaction that creates these new bitcoins and assigns them to the miner is one of the transactions that the miner writes in the block, such that this transaction is part of the Merkle tree contained in each block[4]. Further, the transactions that are made by bitcoin wallets and that are sent to the so-called mempool for the miners to pick them and include them in their blocks can contain a fee that goes to the miner who includes them in a valid block. Hence, once a miner finds a valid block, she receives the newly created bitcoins and the fees of the transactions that she includes in her block.

Figure 2 illustrates the creation of a block. In Figure 2 three miners are competing to find the next valid block—namely, Miner A, Miner B, and Miner C. The last valid block found in this example is Block 8. Each of the three miners uses the hash of Block 8, together with the target, the version, and the Padding + Len, in order to write a potentially valid block. A potentially valid block is a block that fulfills all the requirements to be a valid block with the exception of the nonce. The potentially valid blocks written by each miner are different from each other because despite the fact that the target, the hash of the previous block, and the version and the Padding + Len are the same for all, each miner writes a different timestamp, and includes a different set of transactions (different Merkle root) on it. Once each miner has written a potentially valid block, she tries different nonces until the hash of the hash of a version of the potentially valid block starts with the required number of zeroes. Once this occurs, the hash of this block (Block 9 in our example) becomes, together with the target, the version, and the Padding + Len, the common basis for all the miners to write (each of them) their next potentially valid block (Block 10 in our example).

Mining is a random process. Miners take the last valid block, use it to write a new block that contains the required information, include a nonce in the block, and hash the block twice, hoping to get a valid hash (with at least as many zeroes as required by the target). If miners don't succeed in this process, they increment the nonce by one, add it again to the block, and hash that new block twice trying to get a valid hash at this new attempt. This process is repeated continuously until a hash less than the target value is found by any participating miner. Miners therefore follow a stochastic process to find valid blocks, in which they write a block and then start hashing the block with different nonces or superNonces until a

[4]The amount of the reward started at 50 bitcoins and is halved every 210,000 blocks. This occurs approximately every four years.

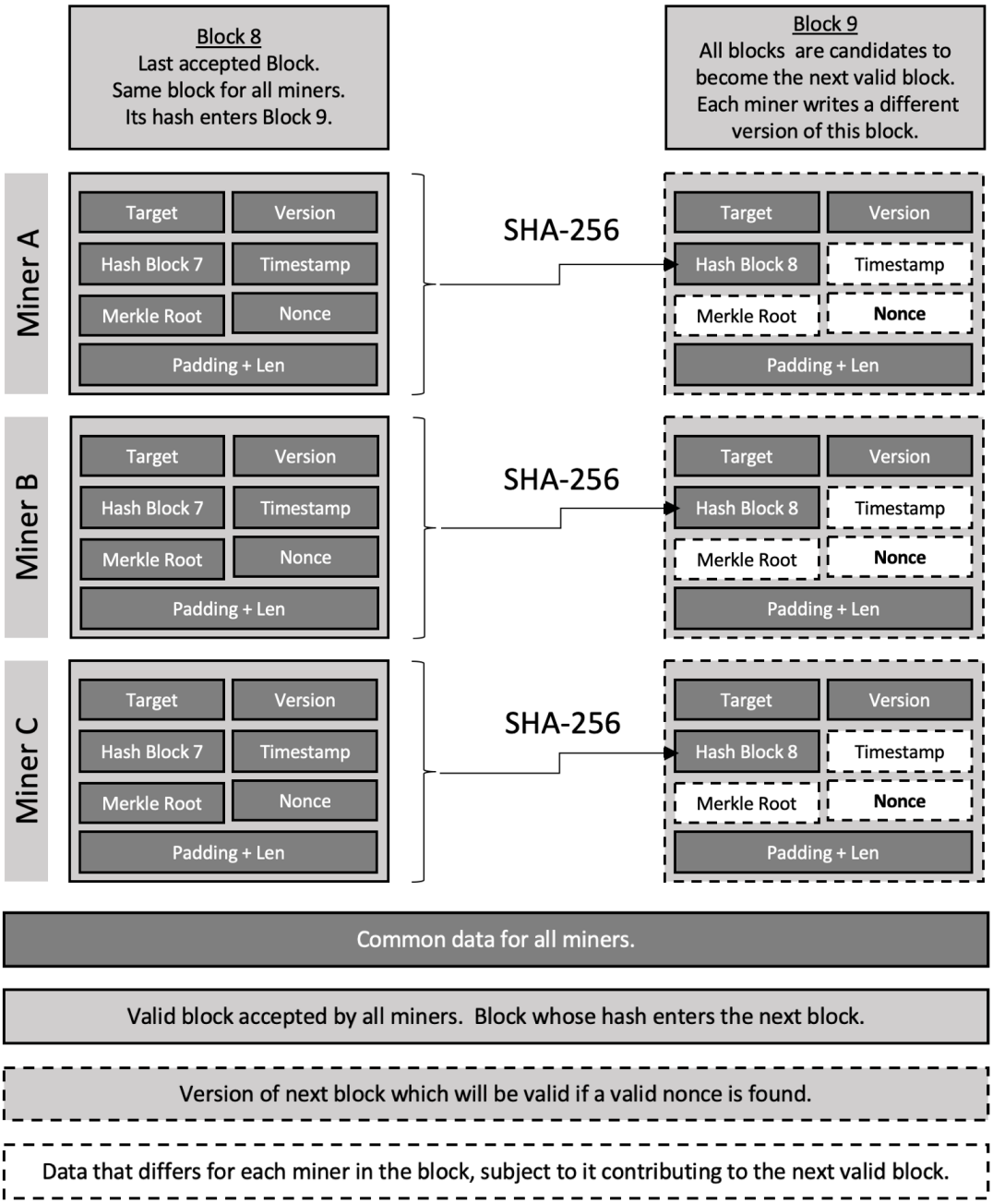

Figure 2: Illustration of the blockchain and the different versions of a block that can become the next valid block.

valid block is found. Miners don't try the same nonce with the same block twice since they already know that it results in an invalid hash.

2.3 Bitcoin Mining as a Poisson Process

Nakamoto (2008) introduces bitcoin and the process of creating new units of the cryptocurrency by hashing the previous block and comparing the result to a certain threshold. He suggests—though he does not explicitly state—that valid blocks are found according to the Poisson distribution. Probability events that follow the Poisson distribution can occur n times in an interval. The average number of events in an interval is designated by λ, which is also called the rate parameter. In the Poisson distribution the probability of observing k events in an interval is given by the equation

$$P(k) = e^{-\lambda}\frac{\lambda^k}{k!}.$$

As described by Rosenfeld (2011), the bitcoin protocol sets the target value (the minimum number of zeroes with which a hash will be considered valid) such that every hash has a probability $\frac{2^{16}-1}{D2^{48}}$ of yielding a valid block. For the sake of simplicity, we—like other scholars—approximate this value by $\frac{1}{D2^{32}}$. This implies that the Poisson parameter that determines the probability of winning of miner i over time t expressed in seconds (the probability of finding a valid block after mining for time t) can be written as

$$\lambda_i = \frac{h_i t}{D2^{32}},$$

$h_i t$ being the hash rate of miner i and D the Difficulty parameter. Therefore, a miner i mining at a rate of h_i hashes per second for time t (time expressed in seconds) has an expected rate of winning $\frac{h_i t}{D2^{32}}$. This assumption is used by many scholars to study different aspects of the bitcoin network. Eyal and Sirer (2013) study the incentive structures for selfish mining in the Bitcoin network (a process tried by some miners to leverage the time advantage they have when they find a valid block) and directly assume the Poisson distribution for the arrival rate of blocks. Decker and Wattenhofer (2013) study the propagation of transactions and blocks through the bitcoin network in order to understand the creation of forks. In their analysis they specifically state that the proof-of-work mining process is a Poisson process, in which the time difference between blocks follows an exponential distribution. Rosenfeld (2014) states that while the qualitative nature of bitcoin mining is well understood, there is widespread confusion about its quantitative aspects and how they relate to attack vectors and their countermeasures. He looks at the stochastic processes underlying typical attacks and their resulting probabilities of success, assuming again

that the number of blocks found by miners follows the Poisson distribution. A. K. Miller and LaViola (2014) present a formal model of anonymous, synchronous processes that communicate using one-way public broadcast, showing that the Bitcoin protocol achieves consensus using this model. For their proofs, they assume that the total number of puzzle solutions found by the network (i.e., the total number of valid blocks found by a miner) follows the Poisson distribution. Göbel et al. (2015) study the effect of propagation delay on the evolution of the Bitcoin network, and use a spatial Poisson process model to study the values computed by Eyal and Sirer (2013). Lewenberg et al. (2015) describe that if blocks in a blockchain (bitcoin or others) are created at a high rate compared to their propagation time in the network, many conflicting blocks are created, which can negatively affect the transaction throughput of the system; they also propose an alternative structure to the chain that allows for operation at higher rates. For both their analysis and their solution, they assume the Poisson distribution for the arrival rate of blocks. Houy (2016) deals with the mining incentives in the Bitcoin protocol, which he defines as a speed game between the miners, which differ from each other in terms of the computational power (hash rate per second) with which they try to find a valid block. In his definition of their game, as well as in the analytic search for Nash equilibria, he assumes that the process of finding a valid block can be modelled as a random variable following the Poisson distribution. Cocco and Marchesi (2016) present an agent-based artificial market model of the Bitcoin mining process, which they use to model the economy of the mining process. They define the probability of mining a valid block as the relative hashing power of the miner at every point in time $r_i(t)$ divided by the total hashing power of all the miners in the network at the same time $r_{Tot}(t)$, such that the number of bitcoins, b, that a miner can expect to mine is the probability of winning multiplied with the total reward, B, of newly created bitcoins: $b = \frac{r_i(t)}{r_{Tot}(t)} B$. Note that the probability of winning remains constant over time such that time plays no role in the expected number of bitcoins won. Solat and Potop-Butucaru (2016) propose and theoretically analyze a solution for the bitcoin selfish mining attack. They propose a solution to prevent such attacks by exploiting the Poisson nature of the proof-of-work mining protocol. Beccuti and Jaag (2017) model the bitcoin mining process as a game of imperfect information, in which miners have to choose whether or not to report their success (selfish mining). They show that the game has a multiplicity of equilibria and that the minimum requirement to find it optimal not to report a block decreases with the number of "selfish" miners. For their analysis, they assume that success in bitcoin mining follows the Poisson distribution with the parameter proposed by Rosenfeld (2011). Chiu and Koeppl (2017) study the optimal design of cryptocurrencies and assess quantitatively how well such currencies can support bilateral trade. They propose a design for cryptocurrencies that reduces mining and relies exclusively on money growth rather than transaction fees to finance mining rewards. For the analysis of the optimality of their design they assume that the probability of a miner finding a valid block is proportional to the fraction of computational

power that that miner owns, utilizing the results of Rosenfeld (2011). Dimitri (2017) presents a game-theoretic framework, assuming complete information in order to model Bitcoin mining. Among other findings, he finds that in the unique pure strategy Nash equilibrium of the game the optimal amount of energy consumption of miners depends on the reward for solving the puzzle (finding a valid block). For the definition and the analysis of the game he assumes the same probability of winning as Rosenfeld (2011). Aggarwal et al. (2018) investigate the exposure of Bitcoin, and other cryptocurrencies, to attacks by quantum computers and suggest that the proof-of-work mining protocol used by Bitcoin is relatively resistant to the threat posed by quantum computers' substantial speedup, mainly because specialized ASIC miners are extremely fast compared to the estimated clock speed of near-term quantum machines. For the success rate of mining they take a probability of the same form as Rosenfeld (2011). L. Cong et al. (2018) provide a dynamic asset-pricing model of cryptocurrencies. For their model, they define the law of motion of token supply (the result of the mining process) as an exogenous stochastic Markov process. They also present an alternative formulation in which the token supply follows the Poisson distribution, as "seen in Bitcoin's supply schedule". They state that formulating the process as a Poisson process has the advantage that equilibria between two Poisson arrivals still have only one state variable, which allows the authors to solve the model by backward induction, starting from the asymptotic future where token supply has plateaued and moving back sequentially in the Poisson time. Hayes (2019) proposes and tests a cost-of-production model for valuing bitcoin. In order to calculate the expected number of bitcoins that a miner can produce (which is crucial for the miner to decide whether she participates in the mining process or not), he implicitly assumes the Poisson distribution for the arrival rate of blocks. Easley et al. (2019) investigate the role that transaction fees play in the Bitcoin blockchain's evolution from a mining-based structure to a market-based economy. They do so by developing a game-theoretic model to explain the factors leading to the emergence of transactions fees, as well as to explain the strategic behavior of miners and users. Their model also highlights the roles played by mining rewards and by volume, and examines how microstructure features such as exogenous structural constraints influence the dynamics and stability of the Bitcoin blockchain. In order to do so, they assume that, independently, for each miner working on the problem, valid blocks arrive according to the Poisson distribution. Wang et al. (2019) provide a systematic vision of the organization of blockchain networks. By emphasizing the unique characteristics of incentivized consensus in blockchain networks, their review of the existing consensus protocols is focused on both the perspective of distributed consensus system design and the perspective of incentive mechanism design. They also assume the arrival rate of blocks to follow the Poisson distribution. L. W. Cong et al. (2019) study how the rise of centralized mining pools for risk sharing affect the relationships between competing miners and the energy consumption of proof-of-work-based blockchains. They state that in proof-of-work mining the probability of finding a solution is not affected by the number of trials previously attempted.

This is what they call the "well-known 'memoryless' property" of proof-of-work mining, which "implies that the event of finding a solution is captured by a Poisson process with the arrival rate proportional to a miner's share of hash rates globally" as described by Eyal and Sirer (2013) and Sapirshtein et al. (2015). By assuming that the arrival rate of blocks follows the Poisson distribution, all these authors are assuming (as nicely stated by L. W. Cong et al. (2019)) that mining is a "memoryless" process, and therefore that the probability of finding a valid block is independent of the previously made attempts.

Grunspan and Perez-Marco (2017) correct the analysis given in Nakamoto (2008) regarding the success of double-spend attacks on the bitcoin blockchain, and give a closed-form formula for the probability of success of a double-spend attack. In doing so, they assume that the number of blocks $N(t)$ mined at time t follows the Poisson distribution. Grunspan and Perez-Marco (2017) do not revise or study the arrival rate of blocks, but demonstrate that one of the characteristics of the bitcoin blockchain that emerges from this arrival rate—namely, the success of double-spend attacks—differs from what was previously assumed in the literature. Their work is especially important for us, since while it still accepts that the arrival rate of blocks follows the Poisson distribution, it challenges the assumption implied by Nakamoto (2008) that the success of a double-spend attack also follows the Poisson distribution.

Bowden et al. (2018) challenge the assumption of the block arrival rate following the Poisson distribution, and based on a stochastic analysis of the block arrival process demonstrate that this is not the case. They present a refined mathematical model for block arrivals, focusing on both the block arrivals during a period of constant difficulty and how the difficulty level evolves over time. Their work leaves the question of "how does the arrival rate of blocks really behave?" open.

2.4 Bitcoin Mining as a Negative Hypergeometric Process

In this section we explain why we consider that the arrival rate of blocks follows the negative hypergeometric distribution, make an analogy between bitcoin mining and the classical urn problem used in statistics, and show the descriptive statistics that have led us to this postulate.

2.4.1 Theoretical Motivation

The negative hypergeometric distribution arises in schemes of sampling without replacement. If in the total population of size N there are M elements that are considered a "success" and $N - M$ elements that are consider a "failure", and we sample elements out of the population without replacement until the number

of "successes" reaches a fixed number m, then the random variable X—the total number of "failures" in the sample—follows the negative hypergeometric distribution $X \sim \text{NHG}\ (N,\ M,\ m)$. The probability mass function (PMF) of a random variable X that follows the negative hypergeometric distribution is given by

$$\Pr(X = k) = \frac{\binom{k+m-1}{k}\binom{N-m-k}{M-m}}{\binom{N}{M}}.$$

Using the negative hypergeometric distribution, and X being a random variable counting the samples number at which the *k-th* success is obtained when sampling without replacement from a set of N objects of which M are considered a success, X follows a negative hypergeometric distribution. In that case, the probability mass function is given by

$$\Pr(X = x) = \frac{\binom{M}{k-1}\binom{N-M}{x-k}}{\binom{N}{x-1}} \times \Big(\frac{M-k+1}{N-x+1}\Big).$$

For more details on the negative hypergeometric distribution see Johnson and Kotz (1969) and G. K. Miller and Fridell (2007).

As we have described, a miner takes an input related to the past history of all transactions as given. Further, she writes a block containing this input, as well as the additionally required data (Merkle tree, timestamp, etc.). Once this information is written in the block, the miner includes a nonce in it and uses all this data together as the input of the SHA-256 function. As a result of this hashing, the miner gets one of the 2^{256} possible 256-bit strings that can result from the SHA-256 function. She then takes the hash resulting from this operation and hashes it again in order to get another of the 2^{256} 256-bit strings, hoping that this second hash starts with the required number of zeroes. Whenever the second hash does not fulfill the requirements of a valid block, the miner changes the nonce in the block and repeats the double-hashing process. Since trying twice a nonce that results in no valid hash would represent a waste of resources, miners do not repeat a nonce with the same block. Therefore, given the relevant information written in a block, a miner tries successive nonces until she or another miner finds a valid block. Should the nonces be exhausted, the miner just includes the previously described "superNonce" in the block and repeats the process. This process is illustrated in Figure 3.

Both the domain and the range of the SHA-256 function are finite. Since miners conduct a double-hashing process to find a valid block, and don't repeat a nonce that is known to fail for a given block, we postulate that the probability of finding a valid block increases with the number of previously tried nonces and, therefore, the arrival rate of valid blocks follows the negative hypergeometric distribution, such that the probability of a miner finding a valid block is not only be dependent on her hash rate but also on the

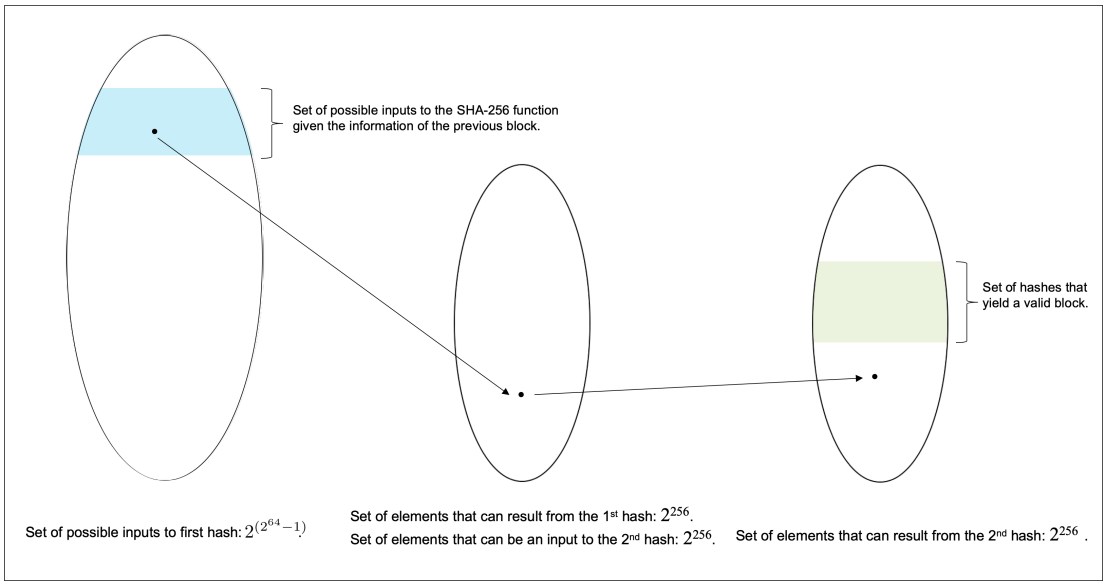

Figure 3: Illustration of the finite domains and ranges of the double-hashing process conducted by bitcoin miners using the SHA-256 function.

number of previously non-valid nonces found for this particular block. In other words, we postulate that while proof-of-work mining is still a "memoryless" process *between* blocks, this does not hold *within* blocks.

Proposition 1 *The number of nonces that needs to be tried until the first valid block is found is a random variable X that follows the negative hypergeometric distribution $X \sim NHG\ (N,\ M,\ 1)$, with N being the number of potentially valid blocks and M the number of potentially valid blocks that, hashed twice, result in a hash below the target.*

In the context of Bitcoin mining, the number of objects with the winning feature equal to M is the number of hashes that start with the required number of zeroes out of all the possible inputs to the SHA-256 function, N, that are contained in the domain of the SHA-256 function[5]. Both N and M are fixed. N equals 2^{256} and M is fixed for each block since the Difficulty, D, that determines it is contained in the header of the block and given by the protocol for each block. The number of draws m made by a miner is the number of nonces tried for a particular block that result in a non-valid hash. No nonce that has previously resulted in a non-valid hash is repeated. The number of observed successes m is set to one since

[5]Given the fact that miners conduct double-hashing, we could argue that N is the range of the SHA-256 function and not its domain. This does not alter our result, due to the fact that the SHA-256 function is deterministic. Further, since the domain of the SHA-256 function is significantly bigger than its range, one could postulate that collisions will occur. In the context of the SHA-256 function a collision would occur whenever two different inputs to the SHA-256 function (two elements of its huge domain containing 2^{64^2-1} elements) yield the same hash (one of the elements in the huge but smaller range of the function, which contains 2^{256} elements). While this is theoretically possible given the fact that the SHA-256 function is non-injective, collisions have not yet been observed. The SHA-256 function is assumed to be surjective but this has not yet been proven.

nonces are tried one after the other and as soon as a valid one is found the problem is solved and miners stop trying to find a valid nonce for their block.

Proposition 2 *The probability of a hash resulting in a valid block after n previously failed hashes, equals*

$$P_n = \frac{1}{D2^{32} - n}.$$

Should this be the case, the probability of finding a valid block would increase with the number of failed attempts for this block, such that the longer it takes to find a block (i.e., the longer the time that has elapsed since the broadcasting of the last block), the higher the probability of success of bigger pools (pools that can conduct more hashes per second) compared to that of smaller pools. This would occur since bigger miners could increase their probability of winning faster than smaller miners, following the PMF described in Proposition 2.

2.4.2 Resemblance between Bitcoin Mining and the Urn Problem

In probability and statistics, an urn problem is a thought experiment in which an event (e.g., success) and its complement (e.g., no success) are represented, respectively, by balls of two different colors (e.g., white balls for success and black balls for no success) contained in an urn. In this thought experiment, an individual draws balls from the urn until she draws a ball of the color she is looking for (white). In the easiest version of the experiment, the individual has two options when drawing a ball of the color that does not represent the event that she is looking for: either return the (black) ball to the urn, or not return it to the urn. In the first case, the probability of drawing a ball representing the event (a white ball in our example) remains constant throughout the many consecutive unsuccessful events due to the fact that the individual always returns a ball representing a lack of success (the black ball in our example) to the urn, leaving the probability of success constant across successive attempts. However, in the second case, the probability of drawing a ball representing the event (a white ball in our example) does not remain constant across the many consecutive unsuccessful events due to the fact that the individual does not return a ball representing a lack of success (the black ball in our example) to the urn, therefore altering the proportion of successful events in the urn after each draw. The first case is called "drawing with replacement" whereas the second case is called "drawing without replacement". In the case of "drawing without replacement", in which balls representing unsuccessful events are not returned to the urn, the probability of drawing a ball representing a success after a fixed number of failures follows the negative hypergeometric distribution. See G. K. Miller and Fridell (2007) for more details about the urn problem and the negative hypergeometric

distribution.

The process of proof-of-work mining resembles a classical urn problem in which an urn contains balls (inputs to the SHA-256 function) that result in either a valid hash (success) or a non-valid hash (failure). Each miner is confronted with such an urn and successively tries inputs (which are in fact nonces for a particular potentially valid block) until she has found a valid one (success). Since nonces known to yield a non-valid hash for a particular potentially valid block are not tried twice, this problem resembles the urn problem "without replacement", in which balls representing a failure event are not returned to the urn. Since miners conduct a double-hashing process, the structure of the problem is slightly more complex, while at the end it simplifies to the classical urn problem due to the fact that the SHA-256 function is deterministic. First, a miner tries a nonce for a given potentially valid block (draws a ball) out of the set of 2^{64^2-1} possible elements in the domain of the SHA-256 function. In the context of the urn problem, she first draws a ball from an urn with 2^{64^2-1} balls. The real space from which the miner samples is nevertheless way smaller since it is restricted to the space that contains a potentially valid block. Out of the urn she can get one of 2^{256} results that are contained in the range of the SHA-256 function. We could say that each of these 2^{256} is a different color (or a number). Regardless of the result of the first hash, the miner hashes the result of the first hashing process and gets the final hash that can be either valid or non-valid. Should the resulting second hash be non-valid, the miner will certainly not try again the same nonce, which she now knows yields a non-valid hash. This is the same as not returning the ball representing the non-success event to the urn.

3 Data

In this section we describe the data used to empirically study our proposition, and present descriptive statistics that motivate our postulate.

3.1 The Bitcoin Blockchain Data

We downloaded all the information about the blockchain available at www.blockchair.com for the blocks 1 to 555,116. These blocks represent the period between January 3, 2009 and December 28, 2018. The raw data contains the following information for each of the blocks: block number, hash of the block, time and date at which the block was mined, the miner (mining pool) that won the block if it is identified, and other metrics about the fees, transactions, and size of the block. An example, for the first and last blocks,

Block Nr	Hash	Miner	Date	Time
1	000...26f	Unknown	2009-01-03	18:15
2	000...048	Unknown	2009-01-09	02:54
...	...	...	...	...
555,116	000...9b8	AntPool	2018-12-28	15:06

Table 1: Blockchain Data

is depicted in Table 1.

Due to the nature of the blockchain, we only observe the miner that wins each block. We can observe neither the hash power of the successful miner nor that of the miners that competed to find a valid version of each block but did not succeed in finding it. Since we need the hash power of the miner in order to relate it to the success of winning a block, we construct a proxy for it in the following manner. First, we add a column to the dataset containing the date (natural day), such that we can observe how many blocks were mined each day. This divides our dataset into 3,637 days and not into 3,646 days (the days elapsed between January 3, 2009 and December 28, 2018) due to the fact that on some days (especially at the beginning) no block was mined. Second, we take a sample of the original dataset, containing only the blocks mined after block 278,310, such that we only take into account the blocks mined from January 2, 2014 onward. Our intention in taking this subset is to leave out of our analysis the initial years, during which Bitcoin was not so well spread and no mining pools were formed, and to only start observing the data after ASIC miners were really accessible to the public. Third, we eliminate all the blocks won by an unknown miner. After conducting these steps, we assume that all the miners that have won a block on one day have tried to mine all the blocks that have been mined on that particular day. This allows us to compute the fraction of daily won blocks of each miner. This fraction is our proxy for the fraction of total hash power that each miner uses in each natural day. This proxy contains noise. Since, however, (big) mining pools remain stable and constantly mine blocks without disconnecting their hardware completely, we consider it a sufficiently valid proxy for the purpose of this analysis.

By way of example, we can consider November 12, 2018 (2018-11-12), a day on which 110 blocks were mined by identified miners (blocks mined by unknown miners were set aside). Table 2 summarizes which miners won how many blocks on that day. We observe that the mining pool "AntPool" mined 22 blocks, whereas the mining pool "BTC.COM" mined 21 blocks, the mining pool "F2Pool" 20 blocks, etc. From these frequencies, we can calculate the proportion of blocks won by each miner on each day. For this example, "AntPool" has a fraction of 0.2, which comes from dividing the number of blocks won by it on that day (22) by the total amount of blocks mined that day (110). We take these fractions as a proxy for the daily fraction of total hash power of each miner. While this proxy does not represent the true value

Date	Pool	Blocks	SIZE	Ranking
2018-11-12	AntPool	22	0.200	1
2018-11-12	BTC.COM	21	0.1909	2
2018-11-12	F2Pool	20	0.1818	3
2018-11-12	SlushPool	14	0.1273	4
2018-11-12	ViaBTC	11	0.1000	5
2018-11-12	BTC.TOP	7	0.0636	6
2018-11-12	BitFury	4	0.0364	7
2018-11-12	Bitcoin.com	3	0.0273	8
2018-11-12	DPOOL	3	0.0273	8
2018-11-12	Bitclub Network	2	0.0182	10
2018-11-12	58COIN	1	0.0091	11
2018-11-12	Eobot	1	0.0091	11
2018-11-12	KanoPool	1	0.0091	11

Table 2: Exemplary data subset for 2018-11-12

of the size of the miner in terms of hash power (the value, in fact, can change on a minute-by-minute or second-by-second basis), and it it assumes that hash power is the only factor explaining success in mining (which is exactly the hypothesis that we want to reject), it is the best proxy we have been able to conceive thus far.

From these fractions, which we will hereafter refer to as "size", we can compute the ranking of the miner in terms of size. We order the miners by their daily size, and for each day we assign the ranking with value 1 to the miner with the highest size, the ranking with value 2 to the miner with the second highest size, etc. Should two miners have the same size (i.e., should we observe a tie), both miners get the smallest of the possible rankings. If a tie occurs, the immediately smaller miner after the tied miners gets her true corresponding ranking value (e.g., where both Bitcoin.com and DPOOL share the same size, both receive the ranking value 8 and the Bitclub Network receives the value 10). Having calculated these values for each miner and each day, we incorporate them into the original dataset, which also contains the time required to mine each block, computed as the difference in seconds elapsed between the time of the previous block and the current block. The resulting dataset contains 256,365 rows, each representing a block, with information about the approximated size of the miner, its size ranking, and the length of the block measured in seconds.

3.2 Descriptive Statistics

Using our dataset, we plot the length of the block in seconds against the ranking of the miner that wins that block. Figure 4 shows the result of this plot. Recall that the biggest miner of the day (i.e., the miner

with the estimated highest hash power on a given day) is the one with the ranking equal to 1. Observing this plot it seems that longer blocks are won by relatively bigger pools (i.e., pools with higher ranking places). In other words, it seems that very long blocks are not won by pools that have a lower hash rate per second and that are therefore relatively smaller. If further statistical analysis confirms this observation, this will imply that there is a "learning" effect within blocks that is proportional to the size of the miner in terms of hash power (i.e., that relatively bigger miners learn faster than relatively smaller miners). One possible explanation for this phenomenon is that the probability of finding a valid block after having found a set of non-valid blocks follows the negative hypergeometric distribution, such that bigger pools' probability of winning increases faster along the length of the block (time in seconds) than smaller pools' probability of winning, due to the fact that relatively bigger miners can try and fail faster (with a higher hash rate per second) than smaller mining pools. Making an analogy with the urn problem, this would mean that bigger pools draw more balls per second without replacement, such that their probability of winning increases with time faster than is the case for relatively smaller miners.

In order to measure this effect, and given the fact that what we consider a "long" block can be arbitrary, we plot—in Figure 5—the probability of the miner with ranking value 1 winning the block given that the block is long. We set different values for what we consider a long block, starting at 575 seconds (the average length of a block) and continuing in steps of 100 seconds until we reach a block time of 4,200 seconds. We observe that the probability of winning of the biggest miner of the day, given that the block is long, tends to increase with the length of the block that we consider "long". This effect becomes visible at a block time of at least 3,275 seconds. In other words, the advantage of the biggest miner starts becoming apparent for blocks of a length equal to or higher than 3,275 seconds. From this length onward, the probability of the miner with ranking 1 winning the block is higher than the probability of the same miner winning any block. These probabilities emerge from the winning frequencies observed in the data and therefore follow no arbitrary decision with regard to what a "long block" is.

These plots motivate our work. Our aim is to learn if what we observe in the plots can be explained by the theory postulated in Section 2.4 or if, on the contrary, the lack of relatively small miners winning long blocks is just explained by the fact that since small miners by their very definition win less blocks, we have not had enough blocks yet to observe small miners winning long blocks. Should we accept the theory that we postulate in Section 2.4, our work could be seen as a tool with which to analyze if the negative hypergeometric distribution that governs the probability of winning blocks can be well approximated by the negative binomial distribution (urn problem with replacement), which is the case when $N, M, N-M \to \infty$, such that $M/N \to p$.

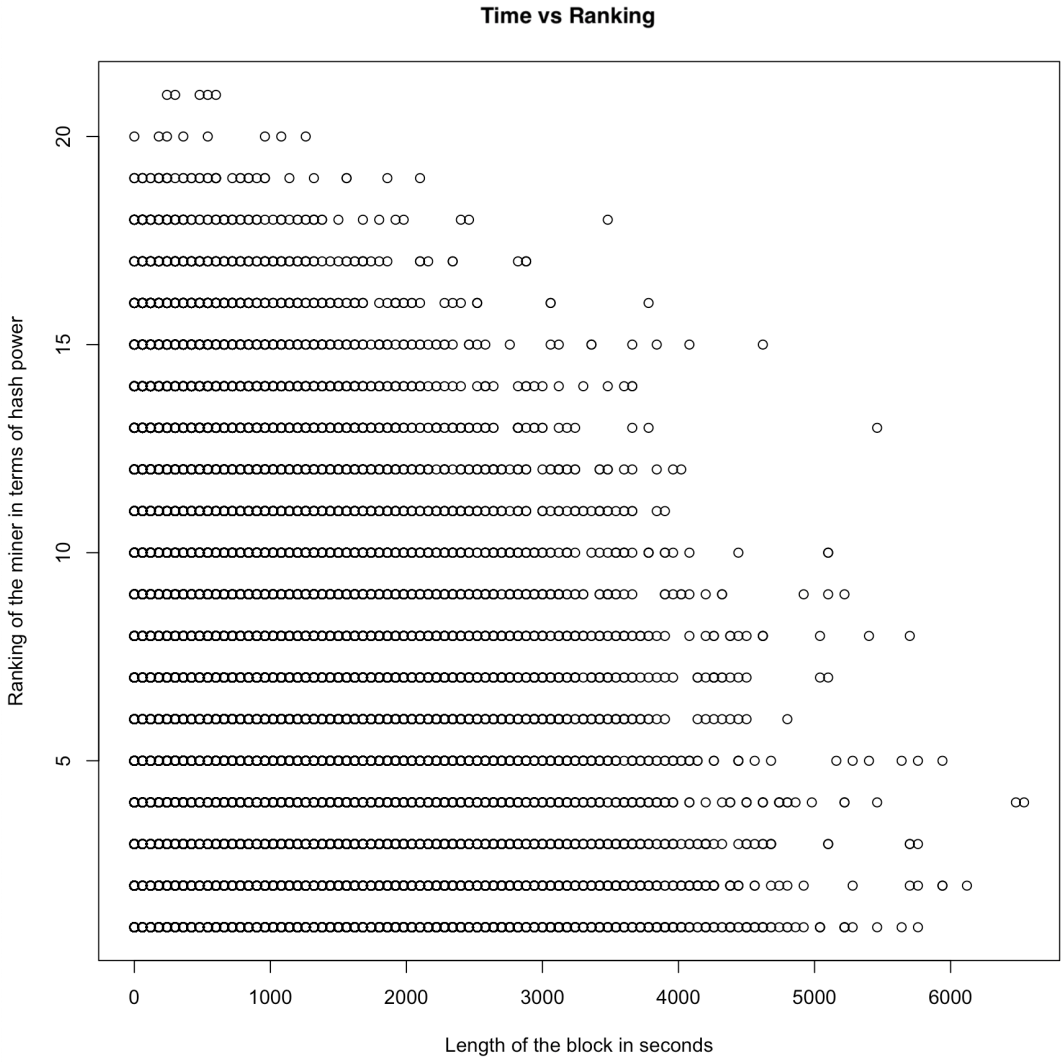

Figure 4: Time required to mine the block vs the ranking of the mining pool in terms of hash power. Period: 2014.01.02 until 2018.12.28.

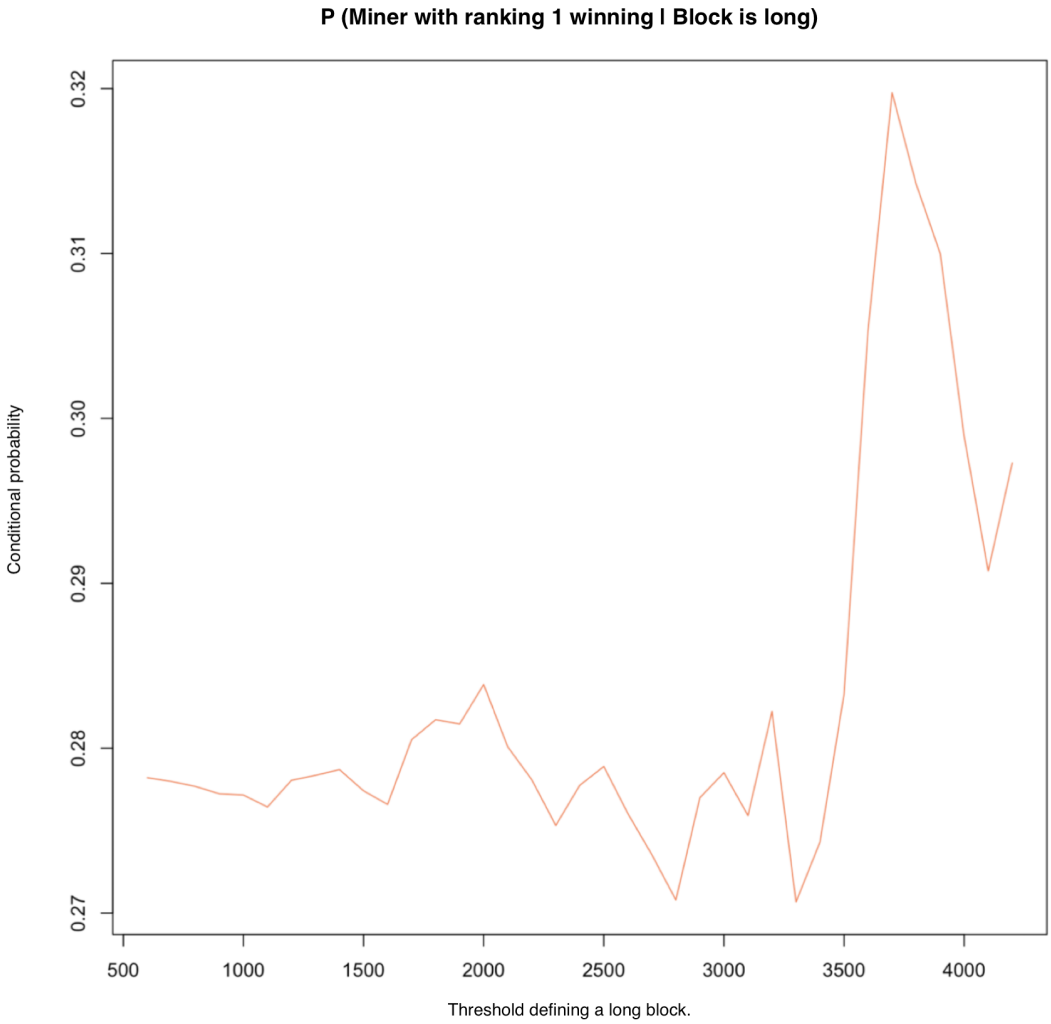

Figure 5: Probability of the biggest miner of the day winning, for different definitions of "long block". Period: 2014.01.02 until 2018.12.28.

4 Statistical Model

Races in which only one of the participants can win have been broadly discussed in the literature. Such races have similarities with uncertain future returns of investments. Authors, such as Snyder (1978), Hausch, Ziemba, and Rubinstein (1981), Bolton and Chapman (1986), and Hausch, Lo, and Ziemba (2008), have investigated the properties of horse race markets to determine the impact of the attributes of the horse and of the race, on the probability of winning for each horse in each race. Bolton and Chapman (1986) present a multinomial logit model to analyze the horse race process, recognizing that only a finite number of mutually exclusive outcomes can occur per race—hat is to say, that one, and only one, of the participating horses wins the race. We model the bitcoin mining process as a race with the same structure as the horse race proposed by Bolton and Chapman (1986), such that we can determine the significant impact of attributes of the miners (their size) and attributes of the race (the time required to mine a block) to find out which of those attributes, individually or combined as interaction variables, have an impact on the probability of winning of a particular miner in a particular block.

4.1 Stochastic Utility Model of the Mining Process

A bitcoin mining race can be understood as an event in which a decision maker—nature—chooses the winning miner out of a pool of all competing miners. In each block, nature is confronted with a choice set consisting of all the miners mining the block. Each miner i has a vector of S attributes associated with it. In our case, this vector contains the size of the miner, and its ranking, as described in Subsection 3.1, denoted $\boldsymbol{x}_i = [x_{i1}, x_{i2}, ..., x_{iS}]$. The adequate parametrization of this choice model requires that the estimated winning probabilities satisfy the standard axioms of non-negative probabilities. Further, the sum of the probabilities of winninf of all the miners needs to be equal to one. The multinomial logit model described below, which has the same structure as the one presented by Bolton and Chapman (1986), fulfills these requirements. The fact that these requirements need to be fulfilled precludes a simple linear regression model for the estimation since it would violate these probability axioms.

We assume the existence of a utility function U that measures the utility derived for nature by each of the participating miners i winning the block. The utility of a miner in each block can be written as

$$U_i = U(\boldsymbol{x}_i).$$

Since there is always an error in the modelling process, which in our case, given the lack of adequate

data, can even be high, the attributes of the vector will not capture all the factors determining the choice. We therefore assume the overall utility of a miner to have two parts. The first part is the deterministic component $V_i = V(\boldsymbol{x}_i)$. The second part is a random component, $\epsilon_i = \epsilon(\boldsymbol{x}_i)$, which captures the errors in the modelling process. If the stochastic error term is independent of its deterministic component, the utility function U can be written as

$$U_i = \boldsymbol{V}_i + \epsilon_i.$$

Since we have a stochastic error term in this equation, our model is a stochastic utility model. Using this model, let us suppose that miner i^* is observed to win a block. This is the same as observing nature choosing miner i^* out of the set of all miners mining a block. Since we are assuming that nature is rational, nature chooses the miner with the highest utility for this block. Revealed preferences imply that $U_{i^*} \geq U_i$ for $i = 1, 2, ..., I$. Given that the utility function is partly stochastic, the probability of miner i^* winning a block can be written as

$$P_{i^*} = Prob(U_{i^*} \geq U_i, i = 1, 2, ..., I).$$

If we assume that the stochastic error terms are identically and independently distributed (i.i.d.) according to the double exponential distribution $Prob(\epsilon_i \geq \epsilon) = exp[-exp(-\epsilon)]$, then the choice probability assumes the closed-form expression of the multinomial logit model:

$$P_{i^*} = \frac{exp(V_{i^*})}{\sum_{i=1}^{I} exp(V_i)} \qquad \text{for } i^* = 1, 2, ..., I.$$

4.2 Estimating the Parameters of the Multinomial Logit Model

The likelihood function associated with a set of blocks can be written for the multinomial logit model as follows:

$$\exp(L) = \prod_{j=1}^{J} P_{jm*},$$

where the subscript j denotes a block ($j = 1, 2, ..., J$), $i*$ is the miner observed to win the block, and L refers to the log-likelihood function.

5 Estimation

In this section we describe the data used for the model's estimation, as well as the specification of the model and its results.

5.1 Data Used in the Model

The computation of the multinomial logit model requires observations of individuals making a choice about an alternative. Both the individual and the alternatives available to the individual can have attributes that explain the choice. In our context the block is the individual and the winning miner is the alternative. We use the attributes of the alternative (the miner), and only those attributes of the individual (the block) that interact with the attributes of the alternative. This is the exact same principle as the one used by Bolton and Chapman (1986). In order to achieve this, we need to use a dataset containing the miner that won the block as well as all the miners that are assumed to have participated in the mining process of a particular block but did not win that block. This allows us to have different alternatives for each (individual) block. For this computation we use the dataset described in Section 3, which contains all blocks won by identified miners for the blocks 278,310 to 555,116.

5.2 Specification of the Model

We use the following form of the multinomial logit model in order to model the utility of each miner m:

$$U_i = \theta_1 SIZE_i + \theta_2 SIZE_i * TIME.$$

The variable SIZE_i is the estimated fraction of daily hash power for a miner that we derive as explained in Section 3.1. We use SIZE_i as a proxy for the miner's size, measured as a fraction of the total hash power of all miners. The miner's size is assumed to be the main determinant of the mining outcome. Further, we use the variable SIZE_i*TIME, which is the interaction of the block length in seconds with the size of the miner. We include this variable in order to measure the impact on the likelihood of winning of the size across time. In this model, the utility for nature of choosing each alternative depends on the attributes of that alternative (the size of the miner), interacted with the attributes of the individual (the length of the block).

A positive and significant coefficient for the variable SIZE_i*TIME would imply that we can reject that the impact of the miner's size on the likelihood of winning a block does not increase over time. In other words, given the significance of the coefficient SIZE*TIME we can reject the hypothesis that the time elapsed since the moment at which the miner starts mining a block does not increase miners' probability of winning in a manner that is proportional to their size. Hence, we would reject that a miner's probability of winning for a particular block does not increases with the number of previously tried and failed potentially valid blocks

(i.e., hashed blocks resulting in a hash above the target). Our postulate would be a possible explanation for this result. Should the coefficient for the variable SIZE_i*TIME be negative and/or non-significant, this would lead to us not being able to reject that the impact of the miner's size on the likelihood of winning a block does not increase over time and therefore that the observations made in Section 3 emerge from the fact that relatively smaller miners have still won too few blocks.

5.3 Results of the Estimation

The model was estimated using the blocks described in Subsection 5.1. The associated empirical results are displayed in Table 1. The results of the estimation show both positive and significant coefficients for the variables SIZE_i and SIZE_i*TIME. The coefficients represent the impact of the variable in the log-odds of each alternative being chosen. The estimate for SIZE is positive with a value of 17.223719 and significant with a p-value of 0. This is obvious and was expected, due to the nature of bitcoin mining and also due to the way in which we have built the proxy for the size. From these results we can infer that ceteris paribus, an increase in the size of the miner (i.e., in her hash power) increases her log-odds of winning. The estimate for SIZE*TIME is positive with a value of 0.0001398 and significant with a p-value of 0.0000638. This implies that given the miner's size, we can reject that her log-odds of winning do not increase with the length of the block (i.e., we cannot reject that the log-odds of winning do not increase with time in a manner that is proportional to the size). Since the positive impact of time (block length) on the log-odds of winning interacts with the SIZE, we cannot reject that the log-odds of winning of a bigger miner do not increase with time in a "faster" or "bigger" manner than those of a smaller miner. Such results reveal that time plays a role in miners' probability of winning. A possible explanation of this phenomenon is Proposition 1 and Proposition 2.

Estimation Results			
Attribute	**Estimate**	**Std. Error**	**P($>\|z\|$)**
SIZE	17.223719	0.0277041	0.0000000
SIZE*TIME	0.0001398	0.0000350	0.0000638

6 Discussion and Conclusion

The motivation to write this piece emerged while we were studying the "entry–exit" problem that miners face when deciding which cryptocurrency to mine with their hardware. This entry–exit problem is governed by the fixed cost of the mining hardware, the variable electricity cost of using the hardware, the expected return in units of each cryptocurrency that can be mined, and the exchange rate of these cryptocurrencies to a fiat currency such as the US dollar. Since the expected return of units of the currency is determined by a distribution that depends on the hash power, we started conducting basic descriptive statistics to better understand our problem. These descriptive statistics—summarized in Section 3—seemed to contradict the assumptions about the Poisson distribution for the arrival rate of blocks in proof-of-work protocols in general and in bitcoin in particular, an assumption that is well established in the literature and that we describe in Section 2. The descriptive statistics suggest that the probability of relatively bigger miners finding longer blocks (longer in terms of the time required to find them) is higher than that of relatively smaller miners. This suggests that there might occur a sort of "learning" when mining a particular block and that relatively bigger miners learn faster than relatively smaller miners. Digging into the literature, we found that recent work by Grunspan and Perez-Marco (2017) and Bowden et al. (2018) has already begun to challenge the assumption of the arrival rate of blocks following the Poisson distribution. However, their respective analyses focus on the security of the proof-of-work protocol and hence on the probability of miners' winning successive blocks. Puzzled by the apparent contradiction between the literature and the statistics emerging from the data, we started revising proof-of-work mining from the basics, postulating that, given a potentially valid block, the number of nonces that needs to be tried by a miner until the first valid block is found is a random variable that follows the negative hypergeometric distribution. This postulate is a possible explanation of the phenomenon observed in the data and is consistent with the technical aspects of bitcoin mining. Further, the resemblance between the urn problem and proof-of-work mining convinces us that this postulate has a theoretical foundation. Should our postulate be correct, it would have serious implications for the way in which scholars and practitioners understand proof-of-work mining. Having postulated this, a question of practical relevance emerged: Does the time that has elapsed since the mining of a block began really play a role in miners' probability of winning?

In order to answer this question, we studied the literature until we found a robust model that would be suitable. Our intention was to measure if the time required to find a block (which represents the number of non-valid nonces tried by a miner for a potentially valid block, given that hash power is measured in hashes per second) had a positive and significant impact on the the likelihood of winning. The model used by Bolton and Chapman (1986) to explain the winning alternative (horse) of a horse race seemed perfectly

suited to answering our question since the structure of our problem is the same as that of their problem. The result of this model applied to our problem indicates that we can reject the hypothesis that the size of a miner (her hash power) does not increase her her odds of winning in a way that is proportional to time (i.e., in a way that is proportional to the time spent computing non-valid versions of a potentially valid block). This suggests that the probability of a miner winning a block varies with the number of previously tried and failed attempts for this block and that, therefore, there exists a sort of "learning" within blocks. Our postulate is a possible explanation for why this is the case.

This result has serious implications for the bitcoin and proof-of-work community. First, it shows that we can still learn more about bitcoin mining. Second, it shows that smaller miners might have an incentive to stop mining a block after trying for a specific time, since after this time their expected reward has decreased so much that it does not compensate their costs. Third, it implies that the way in which platforms are estimating the hash power of mining pools needs to be corrected in order to reflect the impact of time on the success of mining. Fourth, it could be that our postulate explains the concept of mining "luck", since the deviation between the observed and expected performance of miners could be explained by the incorrect approximation of hash power that is used to compute it.

Our results are, though, to be taken with caution. First, since we can only observe the winner of a block, we can only infer which other miners participated in each block. Second, we observe no adequate metrics for hash power and have to derive them from the already inferred and assumed participating miners. Third, the way in which the data is generated implies that the error terms in the model that we use are not i.i.d., which contradicts one of the assumptions of the multinomial logit model.

However, given the theoretical body of our work and the results that emerge from the model—even with a compromised dataset—we conclude that there is enough theoretical and empirical material to sustain that miners' probability of winning does not remain constant over time and that a possible explanation for this phenomenon is that the probability of a miner finding a valid block during a specific attempt follows the negative hypergeometric distribution. This result might have important consequences for the miners of proof-of-work cryptocurrencies in general, and for the miners of bitcoin in particular, as well as for scholars studying cryptocurrencies. We urge the research and practitioner communities to start thinking about mining from this new perspective. Further, we exhort mining pools to report the historical hash power with which they have mined each cryptocurrency, and in a very precise and timely manner. This would allow researchers to continue studying this phenomenon and keep expanding our knowledge of proof-of-work mining.

References

Aggarwal, D., Brennen, G., Lee, T., Santha, M., & Tomamichel, M. (2018). Quantum attacks on bitcoin, and how to protect against them. *Ledger*, *3*(0). Retrieved from https://ledgerjournal.org/ojs/index.php/ledger/article/view/127 doi: 10.5195/ledger.2018.127

Antonopoulos, A. M. (2014). *Mastering bitcoin: Unlocking digital crypto-currencies* (1st ed.). O'Reilly Media, Inc.

Beccuti, J., & Jaag, C. (2017). *The bitcoin mining game: On the optimality of honesty in proof-of-work consensus mechanism* (Working Papers No. 0060). Swiss Economics. Retrieved from https://EconPapers.repec.org/RePEc:chc:wpaper:0060

Bolton, R. N., & Chapman, R. G. (1986). Searching for positive returns at the track: A multinomial logit model for handicapping horse races. *Management Science*, *32*(8), 1040-1060. Retrieved from https://doi.org/10.1287/mnsc.32.8.1040 doi: 10.1287/mnsc.32.8.1040

Bowden, R., Keeler, H. P., Krzesinski, A. E., & Taylor, P. G. (2018). Block arrivals in the bitcoin blockchain. *CoRR*, *abs/1801.07447*. Retrieved from http://arxiv.org/abs/1801.07447

Chiu, J., & Koeppl, T. (2017). *The economics of cryptocurrencies - bitcoin and beyond* (Working Paper No. 1389). Economics Department, Queen's University. Retrieved from https://EconPapers.repec.org/RePEc:qed:wpaper:1389

Cocco, L., & Marchesi, M. (2016, 10). Modeling and simulation of the economics of mining in the bitcoin market. *PLOS ONE*, *11*(10), 1-31. Retrieved from https://doi.org/10.1371/journal.pone.0164603 doi: 10.1371/journal.pone.0164603

CoinMarketCap. (2019). *Coinmarketcap.* Retrieved from https://coinmarketcap.com (Online; accessed 18 March 2019)

Cong, L., Li, Y., & Wang, N. (2018, March). *Tokenomics: Dynamic adoption and valuation* (Working Paper No. 63). Columbia Business School Research Paper. Retrieved from http://dx.doi.org/10.2139/ssrn.3222802 doi: 10.3386/w25592

Cong, L. W., He, Z., & Li, J. (2019, February). *Decentralized mining in centralized pools* (Working Paper No. 25592). National Bureau of Economic Research. Retrieved from http://www.nber.org/papers/w25592 doi: 10.3386/w25592

Courtois, N. T., Grajek, M., & Naik, R. (2014a). Optimizing sha256 in bitcoin mining. In Z. Kotulski, B. Kotulski, & K. Mazur (Eds.), *Cryptography and security systems* (pp. 131–144). Berlin, Heidelberg: Springer Berlin Heidelberg.

Courtois, N. T., Grajek, M., & Naik, R. (2014b). The unreasonable fundamental incertitudes behind bitcoin mining. *CoRR*, *abs/1310.7935*. Retrieved from http://arxiv.org/abs/1310.7935

Decker, C., & Wattenhofer, R. (2013). Information propagation in the bitcoin network. *IEEE P2P 2013 Proceedings*, 1-10.

Dimitri, N. (2017). Bitcoin mining as a contest. *Ledger*, *2*(0), 31–37. Retrieved from `https://ledgerjournal.org/ojs/index.php/ledger/article/view/96` doi: 10.5195/ledger.2017.96

Easley, D., O'Hara, M., & Basu, S. (2019, 5). From mining to markets: The evolution of bitcoin transaction fees. *Journal of Financial Economics (JFE), Forthcoming*. doi: http://dx.doi.org/10.2139/ssrn.3055380

Eyal, I., & Sirer, E. G. (2013). Majority is not enough: Bitcoin mining is vulnerable. *CoRR*, *abs/1311.0243*. Retrieved from `http://arxiv.org/abs/1311.0243`

Göbel, J., Keeler, H. P., Krzesinski, A. E., & Taylor, P. G. (2015). Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. *CoRR*, *abs/1505.05343*. Retrieved from `http://arxiv.org/abs/1505.05343`

Grand View Research. (2018). *Blockchain technology market size, share and trends analysis report by type (public, private, hybrid), by application (financial services, consumer products, technology, telecom), and segment forecasts, 2018 - 2024.*

Grunspan, C., & Perez-Marco, R. (2017). *Double spend races.*

Hanke, T. (2016). Asicboost - A speedup for bitcoin mining. *CoRR*, *abs/1604.00575*. Retrieved from `http://arxiv.org/abs/1604.00575`

Hausch, D., Lo, V., & Ziemba, W. T. (2008). *Efficiency of racetrack betting markets (2008 edition).* World Scientific Publishing Company. Retrieved from `https://books.google.ch/books?id=8ATGCgAAQBAJ`

Hausch, D., Ziemba, W., & Rubinstein, M. (1981, 12). Efficiency of the market for racetrack betting. *Management Science*, *27*, 1435-1452. doi: 10.1287/mnsc.27.12.1435

Hayes, A. S. (2019). Bitcoin price and its marginal cost of production: support for a fundamental value. *Applied Economics Letters*, *26*(7), 554-560. Retrieved from `https://doi.org/10.1080/13504851.2018.1488040` doi: 10.1080/13504851.2018.1488040

Houy, N. (2016). The bitcoin mining game. *Ledger*, *1*(0), 53–68. Retrieved from `http://ledger.pitt.edu/ojs/index.php/ledger/article/view/13` doi: 10.5195/ledger.2016.13

Johnson, N. L., & Kotz, S. (1969). *Distributions in statistics: Discrete distributions.* The Houghton Mifflin Series in Statistics. Boston: Houghton Mifflin Company. XVI, 328 p. $ 12.50 (1969).

Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., & Rosenschein, J. S. (2015). Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems* (pp. 919–927). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. Retrieved from `http://dl.acm.org/citation`

.cfm?id=2772879.2773270

Miller, A. K., & LaViola, J. J. (2014). Byzantine consensus from moderately-hard puzzles : A model for bitcoin..

Miller, G. K., & Fridell, S. L. (2007). A forgotten discrete distribution? reviving the negative hypergeometric model. *The American Statistician*, *61*(4), 347–350. Retrieved from http://www.jstor.org/stable/27643937

Nakamoto, S. (2008). *Bitcoin: A peer-to-peer electronic cash system.* Retrieved from https://bitcoin.org/bitcoin.pdf (Online; accessed 18 March 2019)

Rosenfeld, M. (2011). Analysis of bitcoin pooled mining reward systems. *CoRR*, *abs/1112.4980*. Retrieved from http://arxiv.org/abs/1112.4980

Rosenfeld, M. (2014). Analysis of hashrate-based double spending. *CoRR*, *abs/1402.2009*. Retrieved from http://arxiv.org/abs/1402.2009

Sapirshtein, A., Sompolinsky, Y., & Zohar, A. (2015). Optimal selfish mining strategies in bitcoin. *CoRR*, *abs/1507.06183*. Retrieved from http://arxiv.org/abs/1507.06183

Snyder, W. W. (1978). Horse racing: Testing the efficient markets model. *The Journal of Finance*, *33*(4), 1109-1118. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1978.tb02051.x doi: 10.1111/j.1540-6261.1978.tb02051.x

Solat, S., & Potop-Butucaru, M. (2016). Zeroblock: Preventing selfish mining in bitcoin. *CoRR*, *abs/1605.02435*. Retrieved from http://arxiv.org/abs/1605.02435

Wang, W., Hoang, D. T., Hu, P., Xiong, Z., Niyato, D., Wang, P., ... Kim, D. I. (2019). A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, *7*, 22328-22370. doi: 10.1109/ACCESS.2019.2896108